

Problem 1

If G is not connected, we can analyze each connected component independently, subtracting the combined sizes of all other components from k . Therefore, for the remainder of the solution, we assume that G is connected.

If there exists a subset of vertices X such that $|X| \leq k$ and $G \setminus X$ is a tree, then $\text{tw}(G) \leq k + 1$, as $\text{tw}(G \setminus X) = 1$, where tw denotes treewidth. We will use the algorithm presented in the lecture, which runs in time $27^l \cdot l^{\mathcal{O}(1)} \cdot n^2$, where l is the target treewidth and n is the number of vertices. We apply it to G with $l = k + 1$. The algorithm will yield one of two possible outcomes:

1. Confirmation that $\text{tw}(G) > k + 1$.

In this case, we conclude that no such subset X exists.

2. A tree decomposition of width at most $4k + 8$.

The remainder of the solution focuses on this scenario.

We now proceed with dynamic programming, assuming the decomposition is nice. For any subtree H of the decomposition and its boundary ∂H , let $f : V(\partial H) \rightarrow \{0, \dots, 4k + 9\}$. Define $\text{dp}_H(f)$ as the size of the maximum set $Y \subseteq V(H) \setminus V(\partial H)$ satisfying the following conditions:

1. $Y \cup (V(\partial H) \setminus f^{-1}(0))$ is a forest.
2. Each tree in $Y \cup (V(\partial H) \setminus f^{-1}(0))$ contains at least one vertex from ∂H , unless ∂H is empty (a special case when H is the entire decomposition).

Note that this implies there are at most $|V(\partial H)|$ trees in $Y \cup (V(\partial H) \setminus f^{-1}(0))$.

3. For all $v, u \in V(\partial H) \setminus f^{-1}(0)$, v and u belong to the same connected component (tree) of $Y \cup (V(\partial H) \setminus f^{-1}(0))$ if and only if $f(v) = f(u)$.

To compute dp_H , we consider the following cases:

1. $H = \emptyset$

In this case, we simply set $\text{dp}_H(\text{empty function}) = 0$.

2. $H = \text{introduceVertex}(H', v)$

Here, $\text{dp}_H(f) = \max(\{\text{dp}_{H'}(f') : f = f'[v \mapsto a] \wedge (f^{-1}(a) = \{v\} \vee a = 0)\})$. We use the notation $g[p \mapsto q]$ to denote the function h defined by:

$$h(x) = \begin{cases} q, & \text{if } x = p, \\ g(x), & \text{otherwise.} \end{cases}$$

3. $H = \text{introduceEdge}(H', v, u)$

In this case,

$$\text{dp}_H(f) = \begin{cases} \text{dp}_{H'}(f), & \text{if } f(u) = 0 \text{ or } f(v) = 0, \\ \max(\{\text{dp}_{H'}(f') : f' \text{ is good}\}), & \text{otherwise,} \end{cases}$$

where f' is *good* if it satisfies:

3.1 $f'(v) \neq 0$ and $f'(u) \neq 0$ and $f'(v) \neq f'(u)$.

3.2 For all $w \in V(\partial H)$, if $f'(w) \in \{f'(v), f'(u)\}$, then $f(w) = f'(v)$. Otherwise, $f(w) = f'(w)$.

If no good function f' exists for a given f , set $\text{dp}_H(f) = -\infty$.

4. $H = \text{forgetVertex}(H', v)$

Here, $\text{dp}_H(f) = \max(\{\text{dp}_{H'}(f') + [a \neq 0] : a \in \{0, \dots, 4k+9\} \wedge f' = f[v \mapsto a]\})$, where $[P]$ denotes the Iverson bracket, i.e., $[P] = 1$ if P is true, and $[P] = 0$ otherwise.

Additionally, if a is not zero, we must check whether $f^{-1}(a) = \emptyset$. If this condition holds, we set $\text{dp}_H(f) = -\infty$, except when H represents the entire decomposition, in which case this check is omitted.

5. $H = \text{merge}(H', H'')$

In this case $\partial H' = \partial H''$, and we calculate dp_H as $\text{dp}_H(f) = \text{dp}_{H'}(f) + \text{dp}_{H''}(f)$.

The answer is $\text{dp}_T(\text{empty function}) \geq |V(G)| - k$, where T is the full decomposition.

The total time complexity is bounded by

$$27^{k+1} \cdot (k+1)^{\mathcal{O}(1)} \cdot n^2 + n^{\mathcal{O}(1)} \cdot ((4k+10)^{4k+10})^2.$$

Thus, this algorithm is FPT when parameterized by k , which completes the proof.

Problem 2

We apply the color-coding technique. Let $c = k(l-1)$ denote the maximum total number of vertices along the paths, excluding the start and finish vertices. We color each vertex (excluding the start and finish vertices) with one of c colors. For each $i \in \{1, \dots, k\}$, let $\text{dp}_i(v, C)$ be true if and only if there exists a path from s_i to v such that the set of vertex colors on this path equals exactly C , with no two vertices sharing the same color. Here, we only consider subsets C of size at most $l-1$.

The values of dp_i are initially set to false and are then computed using the following rules:

1. $\text{dp}_i(s_i, \emptyset) = \text{true}$.
2. $\text{dp}_i(v, C) = \bigvee_{(u,v) \in E(D)} (\text{color}(v) \in C \wedge \text{dp}_i(u, C \setminus \{\text{color}(v)\}))$, for all $v \in V(D) \setminus \bigcup_{j=1}^k \{s_j, t_j\}$.
3. $\text{dp}_i(t_i, C) = \bigvee_{(u,t_i) \in E(D)} \text{dp}_i(u, C)$.

To compute these values correctly, ensuring that no dp entry is referenced before it has been calculated, subsets C are considered in increasing order of size. Subsequently, $\text{dp}_i(v, C)$ is filled for all $v \in V(D)$.

The number of such subsets is bounded by:

$$\sum_{j=0}^{l-1} \binom{c}{j} \leq \sum_{j=0}^{l-1} c^j = \frac{c^l - 1}{c - 1} \leq c^l = (k(l-1))^l \leq (kl)^l.$$

Thus, this computation requires $\mathcal{O}((kl)^l k(n+m))$ time, where $n = |V(D)|$, and $m = |E(D)|$.

The number of k -tuples of such subsets is bounded by $((kl)^l)^k = (kl)^{kl}$. We iterate over these tuples, performing the following for each (C_1, \dots, C_k) :

1. Verify that the subsets C_1, \dots, C_k are pairwise disjoint.

This step can be done in $\mathcal{O}(k^3 l)$ time.

2. Check whether $\text{dp}_1(t_1, C_1) = \dots = \text{dp}_k(t_k, C_k) = \text{true}$.

In this case, the answer to the problem is „Yes.”.

The time complexity of one iteration is $\mathcal{O}((kl)^l k(n+m) + (kl)^{kl} k^3 l)$.

The probability of a single iteration failing to find a solution, assuming one exists, is $1 - \frac{c!}{c^c}$, as there are c^c ways to assign c colors to c vertices, and $c!$ of these avoid duplicate colors. Repeating the iteration e^c times reduces the failure probability to:

$$\left(1 - \frac{c!}{c^c}\right)^{e^c} < \left(1 - \frac{1}{e^c}\right)^{e^c} < 1 - \frac{1}{e},$$

a constant.

The total time complexity is:

$$\mathcal{O}(e^{kl}((kl)^l k(n+m) + (kl)^{kl} k^3 l)),$$

which is FPT when parameterized by $k+l$, as $kl \leq (k+l)^2$.

This algorithm can be determinized using an (n, c, c) -splitter, as discussed in the lecture. The splitter, with size $e^c c^{\mathcal{O}(\log c)}$, can be found in time $e^c c^{\mathcal{O}(\log c)} n \log n$. The resulting complexity is:

$$e^{kl} (kl)^{\mathcal{O}(\log kl)} (n \log n + \mathcal{O}((kl)^l k(n+m) + (kl)^{kl} k^3 l)),$$

which is also FPT when parameterized by $k+l$, completing the proof.

Problem 3

Without loss of generality, assume the decomposition is nice. For any subtree H of the decomposition and its boundary ∂H , let $f : V(\partial H) \rightarrow \{1, \dots, t+1\}$ be an injective function. Define $\text{dp}_H(f)$ as the size of the largest set $Y \subseteq E(H)$ such that there exists an ordering σ of the vertices of H satisfying the following conditions:

1. For any edge $(v, u) \in Y$, it holds that $\sigma(v) < \sigma(u)$.
2. For any $v, u \in V(\partial H)$, $f(v) < f(u)$ if and only if $\sigma(v) < \sigma(u)$.

Note that this ordering is not necessarily unique. We only require the existence of at least one such ordering for a given f .

We compute dp_H by considering the following cases:

1. $H = \emptyset$

Here, simply set $\text{dp}_H(\text{empty function}) = 0$.

2. $H = \text{introduceVertex}(H', v)$

In this case, $\text{dp}_H(f) = \max(\{\text{dp}_{H'}(f') : f = f'[v \mapsto a] \wedge f^{-1}(a) = \{v\}\})$.

3. $H = \text{introduceEdge}(H', v, u)$

Here,

$$\text{dp}_H(f) = \begin{cases} \text{dp}_{H'}(f) + 1, & \text{if } f(v) < f(u), \\ \text{dp}_{H'}(f), & \text{otherwise.} \end{cases}$$

4. $H = \text{forgetVertex}(H', v)$

In this case, $\text{dp}_H(f) = \max(\{\text{dp}_{H'}(f') : f' \text{ is good}\})$, where f' is *good* if, for any $u, w \in V(\partial H)$ the condition $f'(u) < f'(w) \iff f(u) < f(w)$ holds.

5. $H = \text{merge}(H', H'')$

Here, $\text{dp}_H(f) = \text{dp}_{H'}(f) + \text{dp}_{H''}(f) - |\{(v, u) : (v, u) \in E(\partial H) \wedge f(v) < f(u)\}|$, since $\partial H = \partial H' = \partial H''$ and we need to subtract the edges that were counted twice.

The final answer is $\text{dp}_T(\text{empty function}) \geq |E(D)| - k$, where T represents the entire decomposition.

The total time complexity is

$$((t+1)^{t+1})^2 \cdot n^{\mathcal{O}(1)} = (t+1)^{2t+2} \cdot n^{\mathcal{O}(1)} = 2^{(2t+2)\log(t+1)} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(t \log t)} \cdot n^{\mathcal{O}(1)},$$

which concludes the proof.